12-20-99

A

Case Docket No. PHA 23,891

THE COMMISSIONER OF PATENTS AND TRADEMARKS, Washington, D.C. 20231

Enclosed for filing is the patent application of Inventor(s):
Farrell L. Ostler; Antoine Farid Dagher

For: BRANCH INSTRUCTIONS WITH DECOUPLED CONDITION AND ADDRESS

ENCLOSED ARE:

[X] Appointment of Associates;
[ ] Information Disclosure Statement, Form PTO-1449 and copies of documents listed therein;
[ ] Preliminary Amendment;
[X] Specification (21 Pages of Specification, Claims, & Abstract);
[X] Declaration and Power of Attorney:
(2 Pages of a [X]fully executed [ ]unsigned Declaration);
[X] Drawing (2 sheets of [ ]informal [X]formal sheets);
[ ] Certified copy of application Serial No. ;
[X] Authorization Pursuant to 37 CFR §1.136(a)(3)
[ ] Other: ;
[X] Assignment to Philips Electronics North America Corporation.

FEE COMPUTATION

CLAIMS AS FILED				
FOR	NUMBER FILED	NUMBER EXTRA	RATE	BASIC FEE - $760.00
Total Claims	8 - 20 =	0	X $18 =	0.00
Independent Claims	3 - 3 =	0	X $78 =	0.00
Multiple Dependent Claims, if any			$260 =	0.00
TOTAL FILING FEE			=	$760.00

Please charge Deposit Account No. 14-1270 in the amount of the total filing fee indicated above, plus any deficiencies. The Commissioner is also hereby authorized to charge any other fees which may be required, except the issue fee, or credit any overpayment to Account No. 14-1270.

[ ]Amend the specification by inserting before the first line as a centered heading --Cross Reference to Related Applications--; and insert below that as a new paragraph --This is a continuation-in-part of application Serial No. , filed , which is herein incorporated by reference--.

CERTIFICATE OF EXPRESS MAILING

Express Mail Mailing Label No. EL335550752US
Date of Deposit December 17, 1999
I hereby certify that this paper and/or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Elissa DeLuccy
Typed Name

Elissa DeLuccy
Signature

Tony Piotrowski, Reg. 42,080
Attorney
(914) 333-9609
U.S. Philips Corporation
580 White Plains Road
Tarrytown, New York 10591

s:\pi\mp15pit0.wa0.doc

BRANCH INSTRUCTIONS WITH DECOUPLED CONDITION AND ADDRESS

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of computer systems, and in particular to processor instruction sets.

2. Description of Related Art

Computer systems utilize "branch" instructions to control program flow. In a conventional computer system comprising a processor and a memory that contains program instructions, the processor executes each program instruction in the memory in sequential order, one instruction at a time. A branch instruction is used to execute program instructions that are located at a specified location in memory, thereby altering the default execution of instructions at sequential memory locations. A conditional branch is used to either continue to process the instructions in sequential order, or to branch to a specified location in memory, depending upon a given condition. The condition is tested, and if the condition is true, the computer processor continues execution at a specified address, otherwise, the processor continues execution at the next sequential location.

Other flow-of-control instructions are also available in most computer systems, such as a subroutine or function "call" and a corresponding "return". (For ease of reference, a function is considered hereinafter as a subset of a subroutine.) As in a branch instruction, a subroutine "call" is used to execute program

instructions that are located at a specified location. As contrast to the branch instruction, a call instruction also causes the processor to store the address of the next sequential instruction after the location of the call instruction (the "return address"). In this manner, a corresponding subroutine "return" instruction will cause the processor to continue execution at the stored return address. Conditional call and return instructions operate similar to the above referenced conditional branch instruction.

The conventional branch and call instructions include two items: an operation-code (op-code) that identifies the instruction as a branch or call instruction, and the address at which the next instruction is located. A conventional return instruction, on the other hand, only includes the operation code. The conventional conditional branch, call, and return instructions are similar to the unconditional branch, call, and return instructions, except that alternative op-codes are used to identify the alternative condition(s) that are used to determine whether to branch to a destination-address or to continue with the next sequential instruction after the conditional instruction.

Typically, an application program contains many branches and subroutine calls, and often branches and subroutine calls to the same destination address are used repeatedly. For example, a "select-case" programming construct allows different segments of program instructions to be executed, depending upon a value of a given parameter, such as a "filing-status" parameter (single, married filing jointly, married filing individually, etc.) in an income tax preparation application program. Within each different

segment of the program corresponding to the value of the given parameter, the same subroutines may be called, albeit with different determined arguments, or with a different combination or order of subroutines, or with a different set of conditions, and so on. Because each conventional branch or call command requires the inclusion of the destination-address, a significant amount of address redundancy exists in a typical program. Subroutine return instructions do not incur this address redundancy penalty, but are limited in use to causing a return from a subroutine call.

The evaluation of conditional instructions in a typical program application can also be problematic. A typical processor includes a set of "flags", and a conditional instruction effects an operation, such as a branch, return, or call, in dependence upon the state of one or more of these flags. Different instructions affect the state of different flags. For example, some instructions affect all flags, some instructions affect no flags, other instructions affect select flags, and so on. A common programming "bug" occurs when a change is made to a program that uses an instruction that affects flags, without an exhaustive assessment of the effects that a potential change in the state of each flag will cause. Because the added instruction may result in the same flag state as the intended flag-setting instruction, this error may not be immediately detected, and the effects of the error may not be associated with this change.

In a typical high-performance computer system, instructions are pre-fetched from memory so that they are available to the processor for execution as soon as the processor is ready to execute the next instruction. In a typical pre-fetch operation, the system pre-

fetches a block of sequential instructions from the system memory into a block of memory that is local to the processor, and therefore immediately available to the processor. When a branch occurs, the previously pre-fetched instructions that sequentially follow the branch instruction in memory are replaced by the instructions from the memory at the specified destination address. This replacement incurs a performance cost while the processor waits for the next instruction from the system memory. Most systems include a "look ahead" capability, wherein the pre-fetch operator looks ahead in the program code being executed to identify the next branch of control. When an unconditional branch is detected, the pre-fetch operator pre-fetches the information from the specified destination-address so that it can provide it to the processor immediately after the branch command is processed. The use of pre-fetching for conditional branch instructions is substantially more complex, because it is unknown whether the next instruction executed after the branch instruction will be the instruction at the destination address or the instruction immediately following the branch instruction. If sufficient local memory is available, segments of instructions from both areas of the system memory are pre-fetched. Alternatively, or in addition to this multiple pre-fetch technique, predictive techniques are often used to pre-fetch the instructions that are likely to be executed. Such predictive techniques are subject to a high degree of variability because in most cases, the state of the condition is virtually unpredictable.

BRIEF SUMMARY OF THE INVENTION

It is an object of this invention to enhance the efficiency of a computer system. It is a further object of this invention to enhance the reliability and maintainability of a program operated on a computer system. It is a further object of this invention to improve the predictive aspects of a pre-fetch system. It is a further object of this invention to ease the task of program development.

These objects, and others, are achieved by providing an architecture that supports the decoupling of parameters typically associated with branch instructions. Branch and call instructions are provided that do not contain an explicit destination address; and, branch, call, and return instructions are provided that do not contain an explicit test condition. In accordance with this invention, the processing system provides a "default" value to any control element in the processor that is not expressly controlled by a particular instruction. In the case of a branch or call instruction, the default destination-address provided to effect the branch or call is the destination-address provided by a prior instruction. Subsequent or alternative branch or call instructions branch to this same address until the default address is set to a different address. In like manner, in most cases, the default condition that is used to determine the result of a conditional test, such as a conditional branch, call, or return instruction, is the last condition specified in a prior instruction. To further support the above objects of this invention, and others, condition-testing can also be effected prior to the execution of conditional instructions. By decoupling the specification of the condition

and/or the destination address from the instructions that cause a branch, call, or return, these instructions can be configured to effect other operations, such as modifying registers or memory elements in preparation for processing by the routine to which the program branches or returns. Also, by providing an explicit means for setting a subsequent default destination address and by providing an explicit means for preevaluating a condition flag, program development and maintenance costs can be expected to decrease, and the variability associated with predicting the outcome and destination-address of conditional branches can also be expected to decrease.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is explained in further detail, and by way of example, with reference to the accompanying drawings wherein:

FIG. 1 illustrates an example processing system in accordance with this invention.

FIG. 2 illustrates an example instruction set format in accordance with this invention.

Throughout the drawings, the same reference numerals indicate similar or corresponding features or functions.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 illustrates an example block diagram of a processing system 100 that provides for a decoupling of conditions and destination addresses from branch instructions, in accordance with this invention. One of ordinary skill in the art will recognize, in

view of this disclosure, that alternative example embodiments can be defined to effect similar results. In the example of FIG. 1, an instruction 110 is provided from a program memory 160; this instruction is decoded by a mapper/default device 140 to provide a microcode instruction 155 that controls the operation of processing circuitry 150. In response to the microcode instruction 155, the processing circuitry 150 effects desired operations on data contained in registers 170 or memory 180. The processing circuitry 150 also determines each next instruction address 151 at each processing cycle, from which each next instruction 110 is obtained from the program memory 160.

As in a conventional processor, the microcode instruction 155 that comprises control bits 155b that control each switch and state device within the processing circuitry 150. That is, for example, the processing circuitry 150 typically comprises a state machine and the microcode instruction 155 provide the input stimuli to this state machine that controls the transition to the next state as well as controlling the production of an output from this state machine. The microcode instruction 155, for example, contains a control bit that will determine whether data will be received from one of the registers 170, whether a location in memory 180 will be written to or read from, and so on. The control bits 155b also determine the operation, such as addition, subtraction, shift, and so on, that will be performed on the data received from the registers 170 or memory 180, and whether other control bits 155b will be affected by the operation. The microcode instruction 155 also includes a "constant" field k 155a. This constant field k 155a is typically

used in a variety of applications. The field k 155a may contain, for example, the aforementioned destination-address of a branch or call instruction, in either absolute form, or relative form. The field k 155a may also contain a constant that is used in the aforementioned mathematical operations on the data contained in the registers 170 or memory 180.

In this example embodiment, as in most processor embodiments, there are fewer bits in the instruction 110 than control elements 155a and 155b in the microcode instruction 155. Conventionally, each instruction is formatted to contain sufficient information for effecting that instruction. In particular, in conventional processor designs, a branch or call instruction contains an explicit designation of a destination address, either as an absolute address, or as an offset, or relative address, relative to the current instruction address 151.

In accordance with the principles of this invention, the mapper/default device 140 provides default values for control elements 155a and 155b that are not explicitly controlled by the current instruction 110. In this manner, prior instructions can be configured to specify such items as the destination-address of a branch or call instruction, the condition to be tested for a conditional branch, call, or return instruction, and so on. In effect, because the mapper/default device 140 allows a destination-address from a prior instruction to be saved for use in a subsequent instruction, the mapper/default device 140 comprises a "default-destination-address" register that is accessible by any instruction that does not contain an explicit destination-address. In like

manner, the mapper/default device 140 in accordance with this invention provides a "condition" register that is accessible by any instruction that does not contain an explicit condition specification.

FIG. 2 illustrates an example of a set of eight different instruction formats 201-208 for use in an example embodiment in accordance with this invention. The format field 210 of each instruction identifies its corresponding format 201-208. Format 201 illustrates an example format that is well suited for data processing operations. Copending U.S. Patent Application "Simple Algorithmic Cryptography Engine", U.S. serial number _____, filed _____ for George Fleming, Farrell Ostler, and Antoine Dagher, Attorney Docket _____ (Disclosure 701049), incorporated by reference herein, discloses a processing system that allows for multiple operations within a single instruction cycle. Consistent with the referenced patent application, select registers 170 are used to address the memory 180, as illustrated in FIG. 1. The instruction format 201 allows for an addressed memory element to be written to or read from, via the Mem 212 field; it also allows for a variety of arithmetic and logical (shift) operations to be performed on the retrieved memory element, via the DataOp 213 field; and, it allows addressed registers to be updated, such as load, increment, decrement, or shift, via the A-update 214 and B-update 215 fields. That is, in accordance with the referenced patent application, data can be retrieved from memory, operated upon, the result stored in a register, and another register incremented, all in the same instruction 110.

Instruction format 202 corresponds substantially to a conventional branch or call instruction, hereinafter termed a "jump" operation. The field "K11" 223 of the format 202 allows for a constant, in this example, a constant of 11 bits, that is used to specify a destination address. The field "jump" 224 defines the type of jump: call, branch, return, relative, absolute, and so on, and the field "cond" 225 defines the condition, if any, that controls a conditional jump. If the condition is true, the next instruction address (151 in FIG. 1) is the address contained in the K11 field 223, zero extended; otherwise the next instruction address 151 is the next sequential address after the address of the current instruction 110. In accordance with the reference patent application, the instruction format 202 also allows a memory operation (store, retrieve) during the same instruction cycle, via the "mem" 222 field, thereby facilitating, for example, the retrieval of a data item in preparation for the commencement of the routine at the destination-address, or a storage of a data item at the end of a cycle controlled by a conditional branch.

In accordance with this invention, the instruction set includes a format 203 that allows for the inclusion of a jump 234 field without an explicit specification of a destination-address. By allowing a jump without an explicit specification of a destination-address, the instruction can contain other operations, such as the DataOp 233 field that controls the operation on data items, as discussed above with regard to the DataOp 213 field. By providing the DataOp 233 field in the same instruction that includes a jump 234 field, a new data value can be computed as a preparation for the

procedure being jumped to via the field 234, or, for example, a function value can be computed and stored in the memory 180 (FIG. 1) upon a return jump from a subroutine.

The format 204 provides additional capabilities in parallel with a jump operation. As illustrated, the format 204 includes a B-update 245 field, similar to the B-update 215 field, in lieu of a condition field. This allows a select register to be updated in parallel with a jump operation. In addition to preparing a data item for the subsequent jump, or storing a computed data item upon return from a subroutine, this update field 245 can be used, for example, to increment a loop index during the same instruction that branches back to the start of the loop. Additional uses of the availability of parallel operations with a jump command, as provided by the decoupling of the destination-address and/or the condition tests from the jump command, will be evident to one of ordinary skill in the art in view of this disclosure.

Formats 205-208 are further variations of this decoupling. In formats 205 and 208, a condition field 255, 285 is provided, without a corresponding jump field. In this manner, the a default-condition can be specified that controls all subsequent jump operations, until another instruction 225, 235, 255, 285 resets the default-condition. In formats 206, 207, and 208, a constant field 265, 275, 283 is provided that can be used to specify a default destination-address for subsequent jump operations. Depending upon the particular convention utilized, these constant fields can be utilized to provide either an absolute address, or a relative address. In accordance with this invention, the default destination-address that

is specified by each of the fields 223, 265, 275, 283 control the destination of all subsequent jump operations, in conjunction with other fields, until the default destination-address is changed by another instruction 202, 206, 207, 208. Format 207 provides a jump 274 field and the constant field 275 that sets the default destination-address, but the condition that controls the jump is not specified. In each of the formats 203-208, the decoupling of the jump, condition, and address fields allows one or more other fields, such as the above described Data-Op 213 field, the A-update 214, and B-update 215 field, to be included in the instruction, to optimize the number of instructions required to effect a task, and/or to optimize the speed of execution of the task, by allowing for a variety of parallel operations in each instruction.

The illustrated mapper/default device 140 of FIG. 1 provides an example embodiment of an instruction processing technique to support the above decoupled operations. As illustrated in FIG. 1, each bit of the instruction 110 is mapped to a corresponding control element in the microcode instruction 155. The mapper/default device 140 comprises a plurality of selectors 141-149 that route each bit of the control field 110b to a corresponding control bit 155b, depending upon the given format 110a of the instruction 110. The mapper/default device 140 routes the different fields from the same bit position of an instruction 110 to different control elements 155a, 155b of the microcode instruction 155, depending upon the format 110a of the instruction 110. The mapper/default device 140 also appropriately controls the other control elements of the microcode instruction 155, those that are not controlled via the

mapping from the instruction 110, by providing default control values for each control element 155a, 155b of the microcode instruction 155. That is, for example, the example instruction formats 201, 203 do not include a constant field for setting the value of k 155a in the microcode instruction 155. The default interpretation for the absence of a specified constant field in an instruction 110 in a preferred embodiment is a null operation. That is, in the absence of a specified constant value, the value k 155a remains the same. Alternatively, as illustrated in format 206 of FIG. 2, when a short, 6-bit value of k is provided in a field K6 265, the default value for the k 155a control element is a sign-extended value of the K6 265 field. A subsequent relative jump operation, via, for example, the jump 234 field of format 203, can use this signed value k 155a to compute a destination-address by adding this value k 155a to the current instruction address 151.

In a preferred embodiment, the output of the multiplexer/selector 141-149 associated with each control element 155 is dependent upon the content of the format field 210, and the inputs are dependent upon the available default options. The choice of a default value or condition for each control element 155a, 155b of the microcode instruction 155 can be any value or condition, but in a preferred embodiment, the default values and conditions are chosen to be those that would be consistent with an assumed value by one of ordinary skill in the art. That is, for example, the zero-ing of higher order bits when a data constant K11 223 is provided, and the sign-extension of higher order bits when a shorter constant K6 265 is provided, would be assumed to be proper defaults by one of

ordinary skill in the art. In like manner, the default condition of the set of control bits corresponding to the absence of a jump 234 field would be assumed to correspond to an advancement of the program counter by one instruction. That is, when an instruction having a format 201 is received, which does not contain a jump 234 field, the default interpretation module 140 sets the appropriate control elements in the macroinstruction 155 corresponding to the absent field 234 such that the corresponding selectors within the set 141-149 are set to select the appropriate inputs to effect an increment of the next-instruction-address 151, thereby providing a consistent, predictable, and logical effect in the absence of an explicit instruction field.

In accordance with another aspect of this invention, the flags 211 field includes control bits that further facilitate the use of decoupled jump instructions. As is common in the art of processing systems, the processing circuitry includes a set of "status flags" that represent the result of particular operations. These status flags include, for example, a "zero" flag that is asserted when the result of a mathematical operation is zero, a "carry" flag that is asserted when the result of a mathematical operation produces an overflow, and so on. One of the flag 211 bits is an "update flags" control bit that effects a preservation of the determined flag values until a subsequent instruction is executed that also has this control bit set. In this manner, when a condition, such as "Branch if Carry" is executed, its result will be determined by the flag that was updated in a prior instruction.

Another control bit in the flags 211 field saves a result of a condition-test into a "saved condition" register, for subsequent use in a conditional jump instruction. The example formats 202, 203, and 205, for example, includes a condition field, 225, 235, and 255, respectively. The condition field 225, 235, 255 contains a specification of the particular condition to test, such as whether the zero flag is set, the carry flag is set, and so on. If the "save condition" control bit is set, the result of the tested condition is saved into the "saved condition" register, for subsequent use in a conditional jump instruction. In a preferred embodiment, whenever the "save condition" control bit is set, the default control bits 155b that control the selection of the condition to utilize for a conditional jump instruction are set to select this "saved condition" register.

By providing one or more flags that maintain their state through other changes of the processor's state, a number of advantages can be realized. As mentioned above, the likelihood of an inadvertent modification of a condition flag during a program update is reduced, because in a preferred embodiment, an explicit setting of the "update flags" or "save condition" control bits in an instruction is required to modify the persistent flag(s). Also, the predetermination of the condition for a conditional branch allows for a highly effective pre-fetch operation, because it is highly likely, and often determinable, that the persistent condition flag will not change before the conditional branch is executed. The presetting of a persistent flag also allows the data item(s) upon

which the condition of the flag is set to change as required without affecting the subsequent conditional instructions.

The foregoing merely illustrates the principles of the invention. It will thus be appreciated that those skilled in the art will be able to devise various arrangements which, although not explicitly described or shown herein, embody the principles of the invention and are thus within its spirit and scope. For example, the principles of this invention can also be effected by providing a separate "destination-address register" that is configured to hold the default destination address, independent of the particular processor architecture utilized. That is, although a microcode and state machine architecture is presented herein as a suitable embodiment to utilize decoupled control conditions, the principles presented herein can be embodied in any processor architecture by including specific devices for storing the decoupled information. These and other system configuration and optimization features will be evident to one of ordinary skill in the art in view of this disclosure, and are included within the scope of the following claims.

CLAIMS

I claim:

1. A processor that is configured to execute program instructions that are stored in a memory, comprising

a default-register that is configured to contain a default-destination-address, and

wherein

the program instructions include:

a first instruction that is configured to cause the processor to load a specified address into the default-register to form the default-destination-address, and

a second instruction that is configured to cause the processor to subsequently execute program instructions that are located in the memory at the default-destination-address contained in the default-register.

2. The processor of claim 1, wherein

the second instruction includes a condition test and is further configured to cause the processor to execute program instructions that are located at the default-destination-address in dependence upon a result of the condition test.

3. The processor of claim 1, wherein

the default-register is further configured to contain a default-condition-test, and

wherein

the second instruction is further configured to cause the

processor to execute program instructions that are located at the default-destination-address in dependence upon a result of the default-condition-test contained in the default-register.

4. The processor of claim 1, wherein

the second instruction is further configured to cause the processor to execute program instructions that are located at the default-destination-address in dependence upon a result of a prior condition-test.

5. The processor of claim 1, wherein

the default-register is further configured to contain a default-condition-test, and

wherein

the program instructions further include:

a third instruction that is configured to cause the processor to execute program instructions that are located at another specified address in dependence upon a result of the default-condition-test contained in the default-register.

6. A processor that is configured to execute program instructions that are stored in a memory, comprising

a default-register that is configured to contain a default-condition-test, and

wherein

the program instructions include:

a first instruction that is configured to cause the processor to load a specified condition into the default-register to form the default-condition-test, and

a second instruction that is configured to cause the processor to subsequently execute program instructions that are located in the memory at a destination-address, based on a result of the default-condition-test.

7. A method of controlling a sequence of program instructions comprising:

executing a first instruction that specifies a destination-address,

executing a second instruction that causes the destination-address to become a next instruction address, and

executing a third instruction that is located at the next instruction address.

8. The method of claim 7, further including

executing a fourth instruction, before executing the second instruction, that specifies a condition-test, and

wherein

causing the destination-address to become the next instruction address is dependent upon a result of the condition-test when the second instruction is executed.

9. The method of claim 7, further including

saving a result of a condition-test, before executing the second instruction, and

wherein

causing the destination-address to become the next instruction address when executing the second instruction is dependent upon the result of the condition-test.

Branch Instructions with Decoupled Condition and Address

ABSTRACT OF THE DISCLOSURE

A processor architecture is provided that supports the decoupling of parameters typically associated with branch/jump instructions. Jump instructions are provided that do not contain an explicit destination address and other jump instructions are provided that do not contain an explicit test condition. In accordance with this invention, the processing system provides a "default" value to any control element in the processor that is not expressly controlled by a particular instruction. In the case of a branch or call instruction, the default destination-address provided to effect the branch or call is the destination-address provided by a prior instruction. Subsequent or alternative branch or call instructions branch to this same address until the default address is set to a different address. In like manner, in most cases, the default condition that is used to determine the result of a conditional test, such as a conditional branch, call, or return instruction, is the last condition specified in a prior instruction. To further support the above objects of this invention, and others, condition-testing can also be effected prior to the execution of conditional instructions.

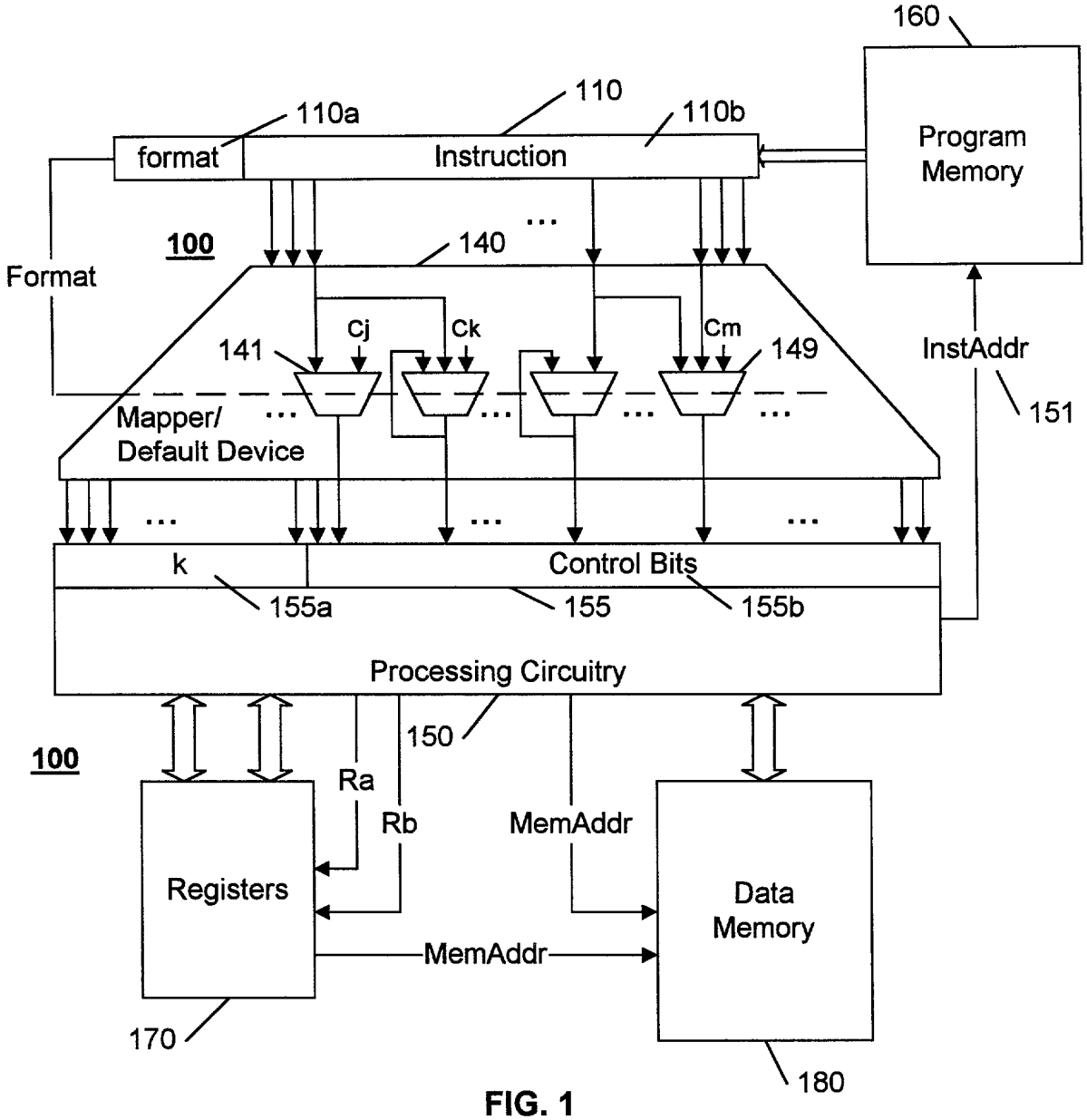
160
Program Memory
110
110a
110b
format
Instruction
100
140
Format
Cj
Ck
Cm
141
149
InstAddr
151
Mapper/
Default Device
k
Control Bits
155a
155
155b
Processing Circuitry
150
100
Ra
Rb
MemAddr
Registers
Data Memory
MemAddr
170
180

FIG. 1

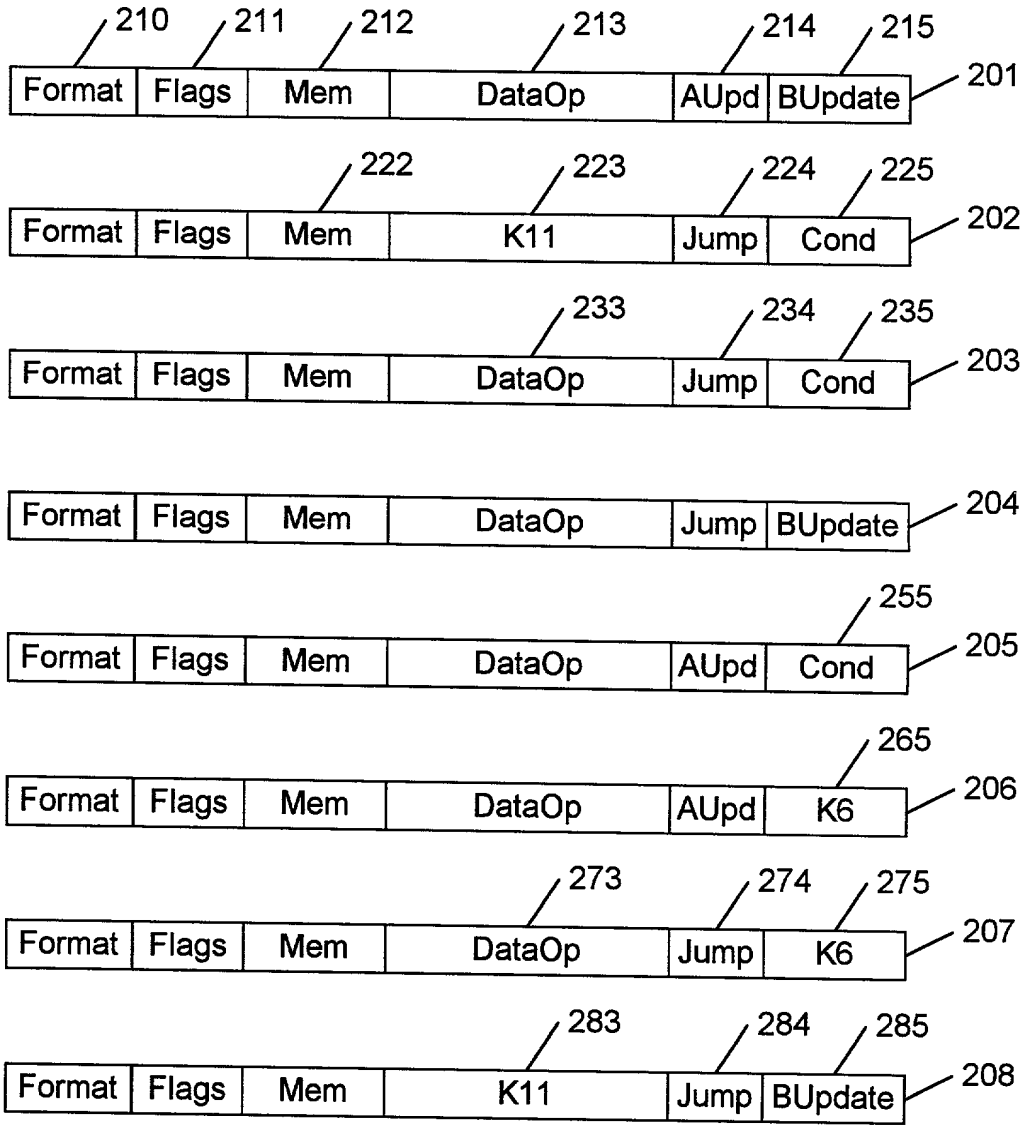
210
211
212
213
214
215
Format
Flags
Mem
DataOp
AUpd
BUpdate
201
222
223
224
225
Format
Flags
Mem
K11
Jump
Cond
202
233
234
235
Format
Flags
Mem
DataOp
Jump
Cond
203
Format
Flags
Mem
DataOp
Jump
BUpdate
204
255
Format
Flags
Mem
DataOp
AUpd
Cond
205
265
Format
Flags
Mem
DataOp
AUpd
K6
206
273
274
275
Format
Flags
Mem
DataOp
Jump
K6
207
283
284
285
Format
Flags
Mem
K11
Jump
BUpdate
208

FIG. 2

DECLARATION and POWER OF ATTORNEY

701046

Attorney's Docket No.
PHA 23,891

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled **"BRANCH INSTRUCTIONS WITH DECOUPLED CONDITION AND ADDRESS'**

the specification of which (check one)

_X_ is attached hereto.

___ was filed on _____ as Application Serial No. _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by the amendment(s) referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulation, §1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

PRIOR FOREIGN APPLICATION(S)

COUNTRY	APPLICATION NUMBER	DATE OF FILING (DAY, MONTH, YEAR)	PRIORITY CLAIMED UNDER 35 U.S.C. 119

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application (s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35 United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

PRIOR UNITED STATES APPLICATION(S)

APPLICATION SERIAL NUMBER	FILING DATE	STATUS (PATENTED, PENDING, ABANDONED)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (list name and registration number)

Algy Tamoshunas, Reg. No. 27,677

Jack E. Haken, Reg. No. 26,902

SEND CORRESPONDENCE TO:	DIRECT TELEPHONE CALLS TO:
Corporate Patent Counsel; U.S. Philips Corporation; 580 White Plains Road; Tarrytown, NY 10591	(name and telephone No.) Tony Piotrowski (914) 333-9609

Dated: 14 December 1999		INVENTOR'S SIGNATURE: Farrell L. Ostler		
Full Name of Inventor	Last Name **OSTLER**	FIRST NAME **FARRELL**	Middle Name **L.**	
Residence & Citizenship	City **ALBUQUERQUE**	STATE OR FOREIGN COUNTRY **NEW MEXICO**	Country of Citizenship **U.S.**	
Post Office Address	Street **9223 MERIWETHER AVE, N.W.**	CITY **ALBUQUERQUE**	State or Country **NEW MEXICO**	Zip Code **87109**

Dated: 14 December 1999		INVENTOR'S SIGNATURE: Antoine Farid Dagher		
Full Name of Inventor	Last Name **DAGHER**	FIRST NAME **ANTOINE**	Middle Name **FARID**	
Residence & Citizenship	City **ALBUQUERQUE**	STATE OR FOREIGN COUNTRY **NEW MEXICO**	Country of Citizenship **U.S.**	
Post Office Address	Street **6709 PLATT PLACE NW**	CITY **ALBUQUERQUE**	State or Country **NEW MEXICO**	Zip Code **87114**

Dated:		INVENTOR'S SIGNATURE:		
Full Name of Inventor	Last Name	FIRST NAME	Middle Name	
Residence & Citizenship	City	STATE OR FOREIGN COUNTRY	Country of Citizenship	
Post Office Address	Street	CITY	State or Country	Zip Code

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of | Atty. Docket

FARRELL L. OSTLER ET AL | PHA 23,891

Serial No. | Group Art Unit:

Filed: CONCURRENTLY | Examiner:

Title: BRANCH INSTRUCTIONS WITH DECOUPLED CONDITION AND ADDRESS

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

APPOINTMENT OF ASSOCIATES

Sir:

The undersigned Attorney of Record hereby revokes all prior appointments (if any) of Associate Attorney(s) or Agent(s) in the above-captioned case and appoints:

Tony Piotrowski (Registration No. 42,080)
(Registration No.) and
(Registration No.)

c/o U.S. PHILIPS CORPORATION, Intellectual Property Department, 580 White Plains Road, Tarrytown, New York 10591, his Associate Attorney(s)/Agent(s) with all the usual powers to prosecute the above-identified application and any division or continuation thereof, to make alterations and amendments therein, and to transact all business in the Patent and Trademark Office connected therewith.

ALL CORRESPONDENCE CONCERNING THIS APPLICATION AND THE LETTERS PATENT WHEN GRANTED SHOULD BE ADDRESSED TO THE UNDERSIGNED ATTORNEY OF RECORD.

Respectfully,

Jack E. Haken, Reg. 26,902
Attorney of Record

Dated at Tarrytown, New York
this December 15, 1999

s:\pi\MP15PIW0.WA0.doc